Constructing CSS Codes with LDPC Codes for the BB84 Quantum Key Distribution Protocol

Maki Ohata* and Kanta Matsuura*
*Department of Information and Communication Engineering
Graduate School of Information Science and Technology
The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo, 152-8505 Japan.
Email: {ohata, kanta}@iis.u-tokyo.ac.jp

arXiv:quant-ph/0702184v3 22 Mar 2007

***Abstract*—In this paper, we propose how to simply construct a pair of linear codes for the BB84 quantum key distribution protocol. This protocol allows unconditional security in the presence of an eavesdropper, and the pair of linear codes is used for error correction and privacy amplification. Since their high decoding performance implies low eavesdropper's mutual information, good design of the two codes is required. The proposed method admits using arbitrary low-density parity-check (LDPC) codes. Therefore, it has low complexity and high performance for hardware implementation. Simulation results show that the pair of codes performs well against practical and various noise levels.**

***Index Terms*—Quantum cryptography, BB84 protocol, quantum key distribution, Calderbank-Shor-Steane codes, low-density parity-check codes.**

I. Introduction

Public-key cryptosystems, based on some computational assumption, such as RSA [18] are becoming weaker due to the progress of cryptanalysis and computing power, although long-term security has been demanded for diverse situations. In addition, if realistic quantum computers appeared, most current public-key cryptosystems would be broken [20].

On the other hand, a one-time pad offers information theoretic security. Due to random outputs that bear no statistical relationship to plaintexts, an adversary cannot obtain information about plaintexts from ciphertexts. A big problem with the one-time pad is that it requires to distribute truly random keys of the same length as plaintexts, at a rate of one per plaintext.

One solution for the key distribution problem was suggested by Bennett and Brassard in 1984 [2]. In the presence of an eavesdropper (Eve) with unlimited computational power, the BB84 quantum key distribution protocol allows unconditional security under the sole assumption that the laws of physics are correct [16], [21].

In the BB84 protocol, a pair of classical binary linear codes, known as Calderbank-Shor-Steane (CSS) codes [4], is used for error correction and privacy amplification over a classical channel when a quantum channel is noisy. With the use of their decoding error probability, Eve's mutual information can be bounded.

Quantum codes, applicable as CSS codes, have been studied previously by employing classical codes such as Hamming, BCH, and Reed-Solomon codes [3], [10]. These classic codes have plainly constructible dual ones for composing CSS codes, but their decoding performance is insufficient to be decodable against practical noise levels over the quantum channel. Consequently, it is necessary to design and evaluate the better pair of codes. In theoretical, after fixing one linear code, randomly choosing the other one is recognized as a good construction approach [23]. However, a non-random and practical method for designing the proper pair of codes remains an open problem.

Low-Density Parity-Check (LDPC) codes [9] are a class of error-correcting linear block codes admitting representations in terms of sparse bipartite graphs, known as Tanner graphs [22], with variable nodes and check nodes. LDPC codes provide near capacity performance on memoryless channels by the sum-product decoding algorithm. The purpose of this paper, therefore, is to construct the good pair of codes using LDPC codes.

MacKay *et al.* have shown how to create the two codes by means of dual-containing LDPC codes [15]. In the dual-containing LDPC codes, every pair of rows in their parity-check matrices must have an even overlap, and every row must have even weight. Because of these properties, the parity-check matrices have many cycles of length four in Tanner graphs. Since short cycles, particularly length four, in Tanner graphs have a bad

influence on decoding performance by the sum-product algorithm, the technique presented in [15] cannot apply optimal LDPC codes. It is suitable for quantum error-correcting codes but not for the BB84 protocol.

In this paper, we propose the simple and practical method for constructing the pair of codes admitting the use of arbitrary LDPC codes. For this reason, our proposal does not need high complexity for hardware implementation, and the pair of codes designed by the proposed method has the same decoding performance as chosen, namely, optimal LDPC codes.

This paper is organized as follows. In Section II we introduce the BB84 quantum key distribution protocol. In Section III we describe efficiently encodable irregular LDPC codes. In Section IV we propose how to construct the two codes using LDPC codes. In Section V we discuss the performance evaluation and show several simulation results for the proposed method. In Section VI we summarize the results.

II. BB84 Quantum Key Distribution Protocol

A. CSS Codes

CSS codes invented by Calderbank, Shor, and Steane [4] are a class of quantum error-correcting codes, and they are derived from classical linear codes by using the concept of dual codes.

We first describe CSS codes. Let C_1 and C_2 be an $[n, k_1]$ and $[n, k_2]$ classical linear code such that $C_1 \supset C_2$. With the use of C_1 and C_2, basis vectors for the CSS code, denoted as an $[n, k_1 - k_2]$ quantum code, subspace can be represented as

$$\boldsymbol{v} \longrightarrow \frac{1}{\sqrt{|C_2|}} \sum_{\boldsymbol{w} \in C_2} |\boldsymbol{v} + \boldsymbol{w}\rangle,$$

where $\boldsymbol{v} \in C_1$. If the code C_2 includes a codeword $\boldsymbol{v}_1 - \boldsymbol{v}_2$, then the codewords corresponding to $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are the same. Thus, these codewords are equivalent to cosets of C_2 in C_1. The code C_1 is used for bit-flip error correction, and the code $C_2^{\perp}$ is done for phase-flip one after the application of an Hadamard transform.

B. Procedure of the BB84 Protocol

In this subsection, for simplicity, we omit the procedure of the BB84 protocol over the quantum channel and introduce that over the classical channel.

In the BB84 protocol, a sender (Alice) and a receiver (Bob) are participants. After communicating over the quantum channel and estimating a noise rate, Alice has an n bit string $\boldsymbol{x}$, and Bob has an n bit string $\boldsymbol{x} + \boldsymbol{e}$. The procedure over the classical channel is as follows:

1) Alice chooses a random n bit codeword $\boldsymbol{u} \in C_1$.
2) Alice announces $\boldsymbol{x} + \boldsymbol{u}$ to Bob.
3) Bob subtracts $\boldsymbol{x} + \boldsymbol{u}$ from $\boldsymbol{x} + \boldsymbol{e}$ and obtains $\boldsymbol{u}'$ by error-correcting the result, $\boldsymbol{u} + \boldsymbol{e}$, to the codeword in C_1.
4) Alice obtains the coset of $\boldsymbol{u} + C_2$ as keys, and Bob obtains the coset of $\boldsymbol{u}' + C_2$ as keys.

The code pair of C_1 and C_2 is CSS codes. In the BB84 protocol, it is used for error correction and privacy amplification. For correcting errors, it requires that the code C_1 can be decodable by some practicable decoding algorithm. Even though Alice's codeword $\boldsymbol{u}$ is not equal to Bob's codeword $\boldsymbol{u}'$, provided that $\boldsymbol{u}' - \boldsymbol{u} \in C_2$, Alice and Bob can share the same keys. Thus, for sharing the keys with high probability, the performance of the coset C_1/C_2 is important rather than that of the code C_1.

C. Security Evaluation of the BB84 Protocol

An entanglement purification protocol can bound Eve's mutual information on the shared keys [13]. If Alice and Bob share a k-EPR-pair state with high fidelity F, where $F > 1 - \delta$, then Eve's mutual information can be bounded by

$$I_{Eve} < -(1-\delta)\log_2(1-\delta) - \delta \log_2 \frac{\delta}{2^{2k}-1}.$$

This inequality shows that high fidelity implies low entropy. By reduction from the entanglement purification protocol to the BB84 protocol via CSS codes [21], the parameter δ corresponds to the decoding error probability of the worse of the two codes $C_1(C_1/C_2)$ and $C_2^{\perp}(C_2^{\perp}/C_1^{\perp})$. Hence, high decoding performance implies low Eve's mutual information, and the decoding error probability of the two codes must be small. In this paper, we analyze the *block* error probability.

III. LDPC Codes

Originally invented by Gallager in the early 1960's [9], LDPC codes have greatly developed as one of the most promising error-correcting codes in the last few years. They have been recently adopted as standard error-control coding techniques in several communication systems.

Every binary linear code of length n and dimension k can be represented as Tanner graphs with variable nodes $v_i, 0 \le i \le n-1$ and check nodes $c_j, 0 \le j \le k-1$. The following is an example for this representation:

Example 3.1: Assume that the parity-check matrix H of LDPC codes of length $n = 6$ and dimension $k = 3$ is

given by

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

The Tanner graph representing this LDPC code is shown in Fig. 1D A cycle of length four can be seen in Fig. 1 that the set $\{v_2, v_3, c_0, c_1\}$ marked by bold lines. Short cycles in Tanner graphs degrade the decoding performance by the sum-product algorithm.

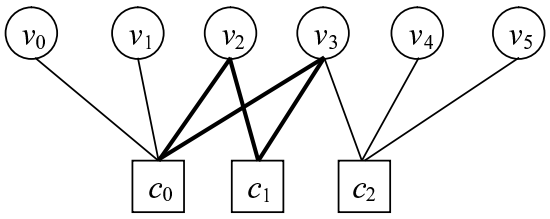

Fig. 1. A Tanner graph

LDPC codes are classified into regular LDPC codes and irregular ones. The decoding performance of irregular LDPC codes is superior to that of regular ones, but irregular LDPC codes usually need high complexity for their encoding processes.

Irregular LDPC codes presented in [8] have algebraic structure and an efficient encoding algorithm. Besides, they perform well as compared to randomly constructed irregular LDPC codes. In this paper, therefore, we choose them as the code C_1.

A. Efficiently Encodable Irregular LDPC Codes

In this subsection, we introduce how to construct irregular LDPC codes proposed by Fujita *et al.* [8]. Let p be an odd prime number such that $2 \leq j \leq k \leq p-1$, and let H be a parity-check matrix defined by an $M(:= pj) \times N(:= pj + pk)$ matrix $[H^{(p)} \mid H^{(d)}]$, where the $M \times M$ submatrix $H^{(p)}$ and the $M \times (N - M)$ submatrix $H^{(d)}$ are defined in block form as follows:

$$H^{(p)} := \begin{bmatrix} T & I & O & \cdots & O & O \\ O & I & I & \cdots & O & O \\ O & O & I & \cdots & O & O \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ O & O & O & \cdots & I & I \\ O & O & O & \cdots & O & I \end{bmatrix},$$

$$H^{(d)} := \begin{bmatrix} I & I & \cdots & I \\ P & P^2 & \cdots & P^k \\ P^2 & P^4 & \cdots & P^{2k} \\ \vdots & \vdots & \vdots & \vdots \\ P^{j-1} & P^{2(j-1)} & \cdots & P^{k(j-1)} \end{bmatrix},$$

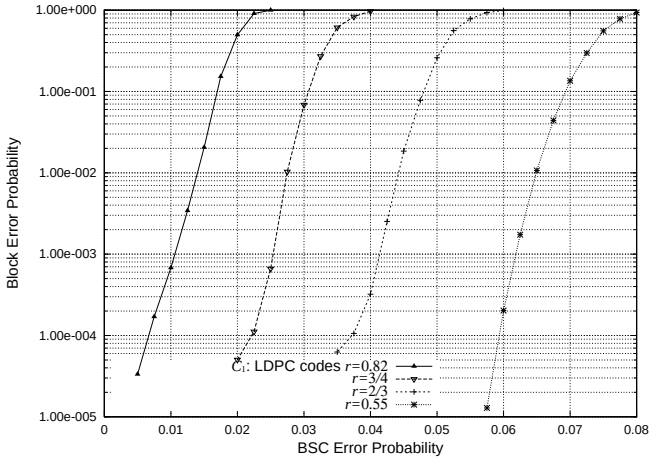

Fig. 2. Decoding performance of irregular LDPC codes of length almost 5000 and several rates on the BSC.

where I is the $p \times p$ identity matrix, O is the $p \times p$ matrix of zeros; the $p \times p$ matrices P and T are defined by the following matrices:

$$P := \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix},$$

$$T := \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

In addition, optimal irregular LDPC codes are constructed by appropriately changing several block component of $H^{(d)}$ into the $p \times p$ matrix O (called a masking method [5]).

Fig. 2 depicts the performance result of almost length 5000 and rate 0.82, 3/4, 2/3, and 0.55 by sum-product decoding with up to 100 iterations on a binary symmetric channel (BSC). In all cases, we employed $p = 73$ as the prime number, and their LDPC codes were designed based on masking matrices given in App. A. Despite the negligible area of probability for error correction, the LDPC codes of rate 3/4 and 2/3 have an error floor starting at the block error probability of 10^{-4} as compared to those of rate 0.82 and 0.55 due to column wight three in their party-check matrices. However, due to little information about the detailed methods for constructing

optimal masking matrices, we note that their elaborate design can further improve their performance.

IV. Design of CSS Codes

In this section, we propose how to construct the pair of codes using LDPC codes. the method is executed by the following procedure:

1) Choose arbitrary LDPC codes defined by an $M \times N$ parity-check matrix H_1 for the code C_1.
2) Separate an $M \times (N - M)$ matrix H_1' from H_1 in ascending order of column weights, encode bits of length $N - M$, where these bits are the row vectors of H_1', to codewords in C_1, and generate the $M \times N$ parity-check matrix H_2 by the set of the codewords of C_1.
3) Defined the code $C_2^{\perp}$ by H_2.

The parity-check matrix H_2 is composed of codewords in C_1, we have $H_1 H_2^T = O$, hence the two codes satisfy the CSS code condition $C_1 \supset C_2$.

This method can apply arbitrary LDPC codes and simply construct the two codes by only choosing the code C_1. By using pseudo-random property of LDPC codes, it pseudo-randomly creates the code C_2. In short, the proposed method is a practical approach of the random construction presented in [23].

The parity-check matrix H_2 has a low-density submatrix and a high-density submatrix. As for theoretical decoding performance, since linear codes defined by high-density parity-check matrices have better performance than those by low-density ones [19], it can be anticipated that the code $C_2^{\perp}$ has a superior decoding property.

Suppose that the rate of C_1 is r, the rate of $C_2^{\perp}$ is also r, so the rate as CSS codes designed by this method is $2r - 1$.

V. Simulation Results

In this section, we show the simulation results for the pair of linear codes constructed by the propose method on the BSC. We can consider errors between Alice and Bob as ones over the BSC. First of all, we discuss how to evaluate the decoding performance of the code $C_2^{\perp}(C_2^{\perp}/C_1^{\perp})$.

A. Decoding Performance Analysis of the Code $C_2^{\perp}$

The code C_1 is LDPC codes; therefore, its performance can be simply evaluated by sum-product decoding. Since the code $C_2^{\perp}$ does not directly correct errors in the BB84 protocol, it is sufficient to analyze the performance by some robust and specifically evaluable decoding algorithm.

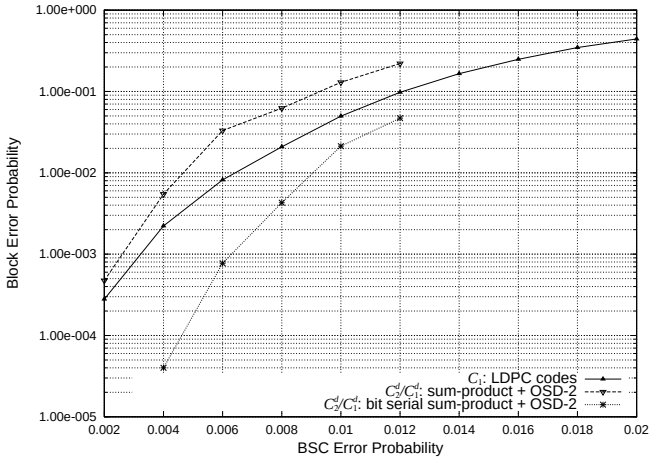

Fig. 3. Decoding performance of the two codes constructed by (3,15) near regular LDPC codes of length 480 and rate 0.8; the rate as CSS codes is 0.6.

While maximum likelihood decoding (MLD) is a powerful algorithm, we cannot evaluate the performance of $C_2^{\perp}(C_2^{\perp}/C_1^{\perp})$ due to its decoding complexity. Moreover, we cannot analyze the performance by the general sum-product decoding algorithm as well because of a high-density part of H_2. Owing to the difficulty of the decoding performance evaluation, both codes tend to be constructed as mediocre LDPC codes and evaluated by sum-product decoding [11], [14], [15].

As a robust decoding algorithm having close performance to MLD, Fossorier *et al.* have proposed to combine sum-product decoding with ordered statistic decoding (OSD) [7]. In the present paper, instead of general sum-product decoding, we applied bit serial sum-product decoding [24], which is the basically same algorithm as shuffled belief propagation decoding [26], and we remove cycles of length four [12], [25] for solving short cycles in Tanner graphs prior to decoding. Bit serial sum-product decoding [24] is the improved algorithm, operating the propagation of likelihood information in Tanner graphs bit-by-bit, of general sum-product decoding, and the algorithm in [12], [25] can remove cycles of length four by transforming Tanner graphs under the code-equivalent condition.

In the first example, we constructed the two codes by (3,15) near regular LDPC codes of length 480 and rate 0.8. The rate as CSS codes is 0.6. Fig. 3 depicts the performance result for C_1 decoded by the general sum-product algorithm, $C_2^{\perp}(C_2^{\perp}/C_1^{\perp})$ done by the original combined algorithm and by the modified one. In both cases of the original combined algorithm and

the modified one, the algorithm of the removal of the cycles of length four is applied prior to sum-product decoding. We chose the maximum numbers of iterations of C_1 and $C_2^{\perp}(C_2^{\perp}/C_1^{\perp})$ to be 100 and 256, respectively. The parameter of the OSD algorithm is set to order-2 reprocessing. In all Figs, henceforth, we note that C^d represents a dual code $C^{\perp}$.

In comparing the original combined algorithm and the modified one, a dramatic improvement with respect to the decoding performance is realized by our modified approach. The decoding performance of $C_2^{\perp}/C_1^{\perp}$ by the modified algorithm after the transformation of the Tanner graph is superior to that of the LDPC code C_1 by sum-product decoding.

B. Approximative Decoding Approach

The modified algorithm has a disadvantage that decoding complexity is too high to decode LDPC codes with block lengths of more than a few thousand bits. Therefore, we suggest and apply an approximative evaluation algorithm. For explaining the algorithm, we give one theorem:

Theorem 5.1: On a binary erasure channel (BEC), sum-product decoding is equal to MLD after the transformation of Tanner graphs (or parity-check matrices) in response to erasure bits.

Proof: When the sum-product decoding algorithm fails on the BEC, a set of erasures is equal to the unique maximum stopping set [6]. The stopping set $\mathcal{S}$ is the subset of the set of variable nodes in the Tanner graph, and all neighbors in $\mathcal{S}$ are connected to $\mathcal{S}$ at least twice. After the weights of the columns having an erasure bit are transformed to one, namely, the number of the connections is one, erasure bits can be decoded by the sum-product algorithm. If they cannot be transformed, the stopping set is a valid codeword. Erasure bits consisting of $\mathcal{S}$ containing valid codewords cannot be decoded even though we exploit MLD. Hence, sum-product decoding is equal to MLD after the transformation of the Tanner graphs in response to erasure bits. ■

High-density parity-check matrices have many small stopping sets due to many cycles, but transforming their Tanner graphs can remove small stopping sets as well as short cycles.

1) Decoding Process: On the basis of Theorem 5.1, we evaluate the performance of $C_2^{\perp}(C_2^{\perp}/C_1^{\perp})$ on the *BSC*, and the detailed procedure is performed as follows:

a) Transform the number of the edges of the nodes having an error in the Tanner graph to less than three.
b) Remove cycles of length four in the Tanner graph.
c) Decode received words to codewords by the general sum-product algorithm.

We anticipate some objections to this methodology, but in our computational simulations, decoding results have no failures when less than a few errors occur in the high-density part of parity-check matrices. To put it more precisely, received words are decoded to codewords in $C_2^{\perp}/C_1^{\perp}$, or they cannot be estimated to any codewords. In all cases that more than a few errors occur in the high-density part of parity-check matrices, no codewords can be estimated due to many cycles in Tanner graphs. For these reasons, only errors in the low-density part of parity-check matrices is important for the evaluation of $C_2^{\perp}(C_2^{\perp}/C_1^{\perp})$, and it is reasonable to suppose that this evaluation algorithm are near optimal on the BSC.

2) Generalized Decoding Process: If we generalize the methodology, then the decoding process is illustrated as follows:

a) Prepare a number of different parity-check matrices, composed of a low-density submatrix and a high-density submatrix, in the same code.
b) Remove cycles of length four in each Tanner graph.
c) Decode received words to codewords by sum-product algorithm using each parity-check matrix.
d) Estimate codewords by a majority decoding result.

In practical, since we cannot prepare numerous parity-check matrices, we assume that the code $C_2^{\perp}$ is decoded by this generalized decoding process and, as mentioned in Sub-subsection V-B1, evaluate the performance by transforming the Tanner graph in response to error bits with one matrix.

C. Decoding Performance of Moderate-Length Codes

In this subsection, we show simulation results for moderate-length codes. As the way for the performance evaluation of $C_2^{\perp}(C_2^{\perp}/C_1^{\perp})$, we applied the above approach.

Figs. 4 and 5 depict the simulation results for the two codes by irregular LDPC codes in [8] of length 2183 and rate 0.78; length 7832 and rate 0.55, respectively. We employed 59 and 89 as the prime number p. Their LDPC codes were designed based on masking matrices given in App. B. In both cases, the decoding performance of $C_2^{\perp}(C_2^{\perp}/C_1^{\perp})$ is superior to that of the LDPC code C_1. Table I summarizes the relationship between the noise and the results covered in $C_2^{\perp}/C_1^{\perp}$ in Fig. 5. We see from Table I that the coset $C_2^{\perp}/C_1^{\perp}$ covers the most codewords that are decoding failures in $C_2^{\perp}$. This is attributed to

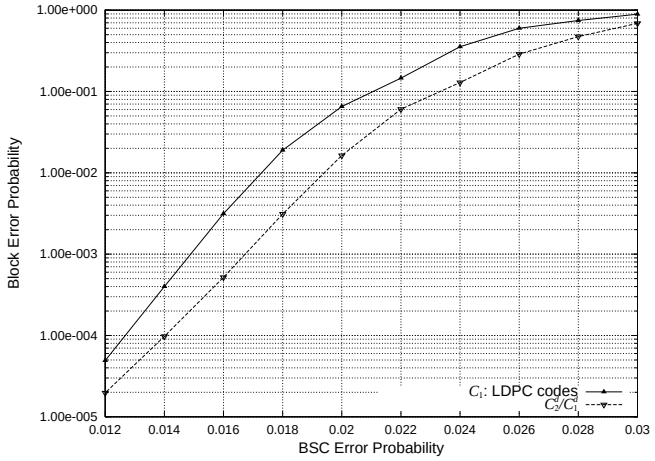

Fig. 4. Decoding performance of the two codes constructed by irregular LDPC codes of length almost 2000 and rate 0.78; the rate as CSS codes is 0.56. The maximum number of iterations was set to 100 and 256, respectively.

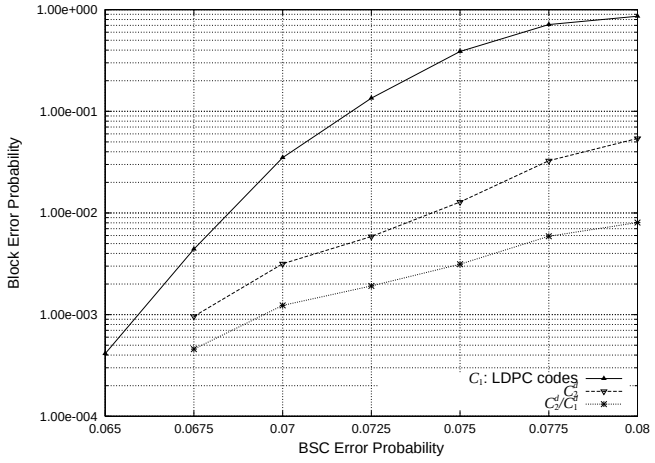

Fig. 5. Decoding performance of the two codes constructed by irregular LDPC codes of length almost 8000 and rate 0.55; the rate as CSS codes is 0.1. The maximum number of iterations was set to 200 and 512, respectively.

TABLE I
SUCCESS RATE COVERED IN $C_2^{\perp}/C_1^{\perp}$ SHOWN IN FIG. 5

Noise rate on BSC (%)	Success rate covered in $C_2^{\perp}/C_1^{\perp}$ (%)
8.0	85.8
7.75	82.4
7.5	75.8
7.25	67.5
7.0	61.1
6.75	52.4

many small codewords in $C_1^{\perp}$, and it results from small row weights in H_1 of low-rate LDPC codes.

Since the code $C_2^{\perp}(C_2^{\perp}/C_1^{\perp})$ has better performance than the LDPC code C_1 in every experimental result, the proposed method can bound Eve's mutual information on keys by the performance of the LDPC code C_1. As for Eve's mutual information on the generated 712 bit key in Fig. 5, for example, they can be bounded by $I_{Eve} < 0.5936$ with the crossover probability 6.5% on the BSC and $I_{Eve} < 6.312$ with the crossover probability 6.75%.

If we choose LDPC codes of almost length 10000 and a little higher rate than 0.5 as the code C_1, then it can be expected that the two codes are decodable with the crossover probability approximately 8.5% on the BSC, keeping Eve's mutual information small.

VI. CONCLUSION

In this paper, we introduced the simple method for constructing the good pair of codes using LDPC codes. First by the combined algorithm of robust decoding and the transformation of Tanner graphs, second by the approximative decoding approach for moderate-length codes, we evaluated the performance. Since the proposed method admits to apply arbitrary LDPC codes, it is less complex and better performance than existing methods. The simulation results indicate that the two codes constructed in this paper can be decodable against practical noise levels on the quantum channel and bound Eve's mutual information on keys by the performance of chosen LDPC codes. Therefore, this method allows secure and effective key generations.

APPENDIX A

Masking matrices W_{rate} used in simulations are as follows:

$$W_{0.82} = \begin{bmatrix}
1000001000000000010001001000001001000001000110011010110111111\\
1010000001000001000100000010001100010010000111011010101110110\\
0000010010001000000100100010010001001000011101101010111011\\
0100010011000001000010000011000100100100011010101010011\\
1001000010001001000100010001000000010001010010111101010110\\
0101001000001000100011000100100000100001000111010000111011\\
0000101010010000100100100100000010000100100100111010101101\\
0010000100010100110001001001000001000110011000101010111110\\
0000100000100110000000100100001010100001001110011111011101\\
0100001000100100011001000000100010001000010011100111101010\\
0001100000010000001000001000011000100001001111111100101111\\
0010100100100010010000101000000010000100010100111111111101
\end{bmatrix},$$

$$W_{3/4} = \begin{bmatrix}
0000001000000100001000000100001010101101011010101011011\\
0000000001000010010100000100001101011010110101101011101\\
1000001000010000001001000000011010110100111010101101110\\
1000010000100000100000100000001011010110011101011011011\\
0000001000001000000100000100101011010110110101101010110\\
0101000010001000000000001000010111010110101011010110110\\
0010000010000100000100000010110101101101101010110010010\\
0001000001000000100001000010001101011010110101101011101\\
1000001000000010001000010000001101010110100101101011010\\
0000100000100001000000100000101101011010110101101011011\\
0100000100000100001000010000010110101101011010110101110\\
0000100000100010000010000100010101101011010110101101101\\
0010000010000010001000001000010101110101101101010110101\\
0000100001000000100000100000101101011010011101011011110\\
0010000100000100001000010000010101101011010110101101101\\
0100010000010000100000100001011101011010110101101011001
\end{bmatrix},$$

$$W_{2/3} = \begin{bmatrix}
0000000100000010100000000000001010100101010010101\\
0000100001000101000000000000010100101010100100101\\
0000000010000001000000001010100101010100101010101\\
0100000000000010000001000010101001010100100101010\\
0000010000100100000000000010101000110100101010100\\
0000000100000000100000000011010100101010010100101\\
0010000001000000001000000100101010010101001010010\\
1000000010000000010000000010101001010101001010101\\
0001000000100000000100001000101001010100101010010\\
0000000000100000000010001100101010010101001010101\\
1001001000001000000000000000010100101010010101010\\
0100000000010000000000011100101001010100101010010\\
0000000000001000001000001000101010010011010001010\\
0000100000010000000010000010010101001010010100101\\
0000010000001000000100000000101001010101010100001\\
0010001000000000100000110100101010010100101001010\\
0000010000000001000001010000100101010010101010101\\
0100000100000100000000000100101010100101001010100\\
0010101000000000000000010000010101010010100101110\\
0001000000100000100001000000101100011001010100101\\
1000000000000100000010101100101001010100101010101\\
0000000010000001000010000001010100101010010101010
\end{bmatrix},$$

$$
W_{0.55} = \begin{bmatrix}
000001000010000000000000000100001010000001 \\
000010000000100000001010000001000100000000 \\
000000001000000000101000000000010001010000 \\
000010000000010001000000010000000000010010 \\
100000000100000000000001100001000000000100 \\
001000100000001000000000000000000100000101 \\
001001000000010000000010000001000100000000 \\
000100000000101000000000000000000101001000 \\
000000100000000000100101000000000000010010 \\
100010000100000000010000000100010000000000 \\
010000000000100000000010010001000100000000 \\
000000000101000010000000000100000010000001 \\
001000000000010001000000100000000000101000 \\
000010000000001010000000000000001010001000 \\
000100000000100000000000100000101000000001 \\
000000001100000010010000000100000000100000 \\
100000100010000000000100100000000010000000 \\
010001000100000000001000000000100000000100 \\
010000000000000010010000000010010010000000 \\
100000000100000001000010000000000100100000 \\
000001000000000100100100000010000000010000 \\
000000110100100000000000000100000000100000 \\
000100000000100000100100100000000100000001 \\
001000001000000000000000000100100000001001 \\
010000000001000000000000010100000000000100 \\
000100000001000000000100000100010000000010 \\
001000000100100100000010000100000000000100 \\
000100010000000100101000000000001000000001 \\
000010000000010010000000010001000000000010 \\
100000010000000100010000000100000000000100 \\
010001010000000100010000000100000000000001
\end{bmatrix}.
$$

Appendix B

Masking matrices W_{rate} used in simulations are as follows:

$$
W_{0.8} = \begin{bmatrix}
0001010101100011101011010101 \\
0010010011011111101101001011 0 \\
1000101001010011011101101111 0 \\
0110001001101001110110110100 1 \\
1001100100111111010001010111 0 \\
0100101000110101011111010110 1 \\
0101000111011100011101110001 1 \\
1010010010101110110010111101 1
\end{bmatrix},
$$

$$
W_{0.55} = \begin{bmatrix}
0010000001000000000000000001000000000000001000000000010 \\
0001000000000000000000000000000000100010010001000001 \\
0100100000000000000100110000000000000000100000000000000 \\
1000000100100000000000000100000010000000000000000000100 \\
0000000000100100000010000000000100000000000000000110000 \\
0000000100000100010100010000000000100000000000000000000 \\
0000000000000000110000000000000100000011000000010000000 \\
00000100000010000000000000001001000000000000001000001000 \\
0001001000000000000000000100000001010010000000000000000 \\
0000000000100000001000000000000000010000001101000000000 \\
0000100000000000100010001001000000000000000000000100000 \\
0000000100000010000000001000011000000000000000100000000 \\
0000000000001000010000000000000000000001100000000010100 \\
0000000000000100000001000010000000000000100001000001 \\
1010000000000000010000100000000010000000000000000001000 \\
0100010011000000000000000000000100000000000000000000010 \\
0000100000000000001000000000010001000010000000000000010 \\
0000000000000000100001000000000000000000011010000001 \\
1011000100000010001000000000000000000000000000000000000 \\
0000001000000000000000000000001000110000000001010000 \\
0000010000101000000010000100000100000000000000000000000 \\
0100000000101000000000000000100000000100000000000100000 \\
0000000000000000100010001000010000000001000000000100 \\
0000000010000000000000001000010000000001000100001000 \\
0001001000000000000000010000100010001000000000000000000 \\
0100000000000010000000000001000000000001000100000000010 \\
0010010000110000000000000010001000000000000000000000000 \\
0000000000000001000000010000000001110000000000000001000 \\
0000100000000000001011000000000010000000000000001000000 \\
00000000100010000000000000000010000000000000000001100001 \\
0000000000100000000100001000000000000000001101000000000 \\
1000000010000000001000000000000000000000000010000000010100 \\
0000000100010001000000000000000000010000010000000100000 \\
0100000101000000000000000000000000000100100000000000001000 \\
00001000000000000001000000000000100000010000001010000000 \\
001000000000000000000100010000000000000000010000000010010 \\
1000000001000000000000001001010000001000000000000000000 \\
0000000000000111000000000000000000000000000100010000001 \\
0001000000001000000000101000000000000000100001000000000 \\
0000010000000000000010000000100110000000000000100000000100
\end{bmatrix}.
$$

Almost all matrices in App. A and B were designed based on degree distribution pairs given in [1].

References

[1] A. Amraoui and R. Urbanke, "LpdcOpt," EPFL, Switzerland, renewal of at any time. Available at `http://lthcwww.epfl.ch/research/ldpcopt/`.

[2] C. H. Bennett and G. Brassard, "Quantum cryptography: public key distribution and coin tossing," *Proc. of IEEE Int. Conf. on Comp., Syst. and Sig. Proc.*, pp. 175-179, Bangalore, India, Dec. 10-12, 1984.

[3] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, "Quantum error correction via codes over GF (4)," *IEEE Trans. Inf. Theory*, vol. 44, no .4, pp. 1369-1387, 1998, *arXiv e-Print* quant-ph/9608006.

[4] A. R. Calderbank and P. W. Shor, "Good quantum error correcting codes exist," *Phys. Rev. A*, vol. 54, pp. 1098-1105, 1996, *arXiv e-Print* quant-ph/9512032; A. M. Steane, "Multiple particle interference and error correction," *Proc. the Royal Soc. of London A*, vol. 452, pp. 2551-2577, 1996, *arXiv e-Print* quant-ph/9601029.

[5] L. Chen, I. Djurdjevic, J. Xu, S. Lin, and K. Abdel-Ghaffar,"Construction of quasi-cyclic LDPC codes based on the minimum weight codewords of Reed-Solomon codes," *Proc. of 2004 IEEE ISIT*, pp. 239, Chicago, IL USA, June 27-July 2, 2004.

[6] C. Di, D. Proietti, T. Richardson, E. Telatar, and R. Urbanke,"Finite length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. IT-48, no. 6, pp. 1570-1579, Jun. 2002.

[7] M. P. C. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics, " *IEEE Tran. Inf. Theory*, pp.1379-1396, Sept. 1995.

[8] H. Fujita, M. Ohata, and K. Sakaniwa, "An algebraic method for constructing efficiently encodable irregular LDPC codes," *Proc. of 2005 IEEE ISIT*, pp. 855-859, Adelaide, Australia, Sep. 4-9, 2005.

[9] R. G. Gallager, "Low-density parity-check codes," Cambridge, MA: MIT Press, 1963; preliminary version in *IRE Trans. on Inf. Theory*, vol. 8, pp. 21-28, Jan. 1962.

[10] M. Grassl, W. Geiselmann, and T. Beth, "Quantum Reed-Solomon codes," *Lect. notes comp. sci.*, vol. 1719, pp. 231-244, 1999, *arXiv e-Print* quant-ph/9910059.

[11] M. Hagiwara and H. Imai, "Design of quasi-regular CSS-LDPC codes by FLWC problem and its application for QKD," *Proc. of SITA 2006*, vol. 2, pp. 679-682, Japan, Nov. 28-Dec. 1, 2006.

[12] K. Kasai, T. Shibuya, and K. Sakaniwa, "A code-equivalent transformation removing cycles of length four in Tanner graphs," *IEICE Tech. Rep.*, vol. 104, no. 302, pp. 25-28, Sep. 2004. (in Japanese)

[13] H.-K. Lo and H. F. Chau, "Unconditional security of quantum key distribution over arbitrarily long distances," *Science*, vol. 283, pp. 2050-2056, 1999, *arXiv e-Print* quant-ph/9803006.

[14] H. Lou and J. Garcia-Frias, "Quantum error-correction using codes with low-density generator matrix," *2005 IEEE 6th Workshop on Signal Proc. Adv. in Wireless Comm.*, pp. 1043-1047, June 2005.

[15] D. J. C. MacKay, G. Mitchison, and P. L. McFadden, "Sparse-graph codes for quantum error-correction," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, Oct. 2004, *arXiv e-Print* quant-ph/0304161.

[16] D. Mayers, "Unconditional security in quantum cryptography," *Jour. of the ACM*, vol. 48, no. 3, pp. 351-406, May 2001, *arXiv e-Print* quant-ph/9802025; preliminary version in D. Mayers, "Quantum key distribution and string oblivous transfer in noisy channels," *Adv. in Cryptology - Proc. of Crypto 1996*, pp. 343-357.

[17] T. Richardson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol.47, pp. 619-637, Feb. 2001.

[18] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Comm. of the ACM*, vol. 21, no. 2, pp. 120-126, Feb. 1978.

[19] I. Sason and R. Urbanke, "Parity-check density versus performance of binary linear block codes over memoryless symmetric channels," *IEEE Trans. Inf. Theory*, vol. 49, no. 7, pp. 1611-1635, July 2003.

[20] P. W. Shor, "Polynomial-time algorithms for prime factorization

and discrete logarithms on a quantum computer," *SIAM Jour. on Computing*, vol. 26, no. 5, pp. 1484-1509, 1997.

[21] P. W. Shor and J. Preskill, "Simple proof of security of the BB84 quantum key distribution protocol," *Phys. Rev. Lett.*, vol. 85, no. 2, pp. 441-444, July 2000.

[22] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, pp. 533-547, Sep. 1981.

[23] S. Watanabe, R. Matsumoto, and T. Uyematsu, "Noise tolerance of the BB84 protocol with random privacy," *Proc. of 2005 IEEE ISIT*, pp. 1013-1017, Adelaide, Australia, Sep. 4-9, 2005.

[24] K. Yamaguchi, Y. Kurihara, M. Yabe, and K. Kobayashi, "Studies on bit serial sum-product decoding based on controlling decoding order," *Proc. of ISITA*, pp. 1001-1006, Parma, Italy, 2004.

[25] J. S. Yedidia, J. Chen, and M. P. C. Fossorier, "Generating code representations suitable for belief propagation decoding," *Proc. of the 40th Annual Allerton Conf. on Comm., Control, and Comp.*, 2002.

[26] J. Zhang and M. P. C. Fossorier, "Shuffled belief propagation decoding,h *Proc. of Asilomar Conf. on Signals, Systems and Comp.*, vol. 1, pp. 8-15, Nov. 2002.